Survey on Implementation of Ant Colony Optimization in Load Balancing

Elnaz Shafigh Fard

Faculty of Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran

Abstract— *Ant colony optimization (ACO) takes inspiration from the foraging behavior of real ant species. This ACO exploits a similar mechanism for solving optimization problems 0for the various engineering field of study especially load balancing.*

In distributed systems, load balancing is one of the central problems that have to be solved in parallel and Grid computation. This paper tries to show some kinds of load balancing techniques for massive computing.

This paper reviews the different load balancing methods includes hardware and software and compares them. Furthermore, it explains their advantages and a disadvantage, showing which method is more adaptive and flexible.

Keywords— Ant colony optimization, load balancing, hardware, software.

I. INTRODUCTION

Grid Computing is a new way of parallel and distributed computing [R.U. Payli, E. Yilmaz, A. Ecer, H.U. Akay, and S. Chien, 2004]. Using grid computing, an individual can unite pools of servers, storage systems and networks into a large system. End users and applications take this environment as a big virtual computing system. The systems connected together by a grid might be in the same room or distributed globally, running on multiple hardware platforms and different operating systems, and owned by different Organizations.

One of the main features of a distributed system [Javier Bustos-Jiménez, Denis Caromel, José M. Piquer, 2007] is the ability to redistribute tasks among its processors. This requires a redistribution policy to enhance productivity by dispatching the tasks in such a way that the resources are used efficiently, i.e. minimizing the average idle time of the processors and improving application's performance. This technique is known as load balancing. Moreover, when the redistribution decisions are taken at runtime, it is called dynamic load balancing.

About load balancing that is one of the most challenging items in traditional distributed system a lot of works has been done. In a number of studies [Casavant and Kuhl, 1994], [Xu and Lau, 1997], [Zaki, Li, and S. Parthasarathy, 1996]. Load balancing mechanisms can be broadly categorized as **global or local , centralized** or **decentralized**, **dynamic** or **static**, and **periodic** or **non-periodic** [Baldeschwieler, Blumofe, Brewer, Atlas,1995].

In a centralized algorithm, there is a central scheduler which gathers all load information from the nodes and makes appropriate decisions. However, this approach is not scalable for a vast environment like the Grid. In decentralized models, there is not usually a specific node known as a server or collector. Instead, all nodes have information about some or all other nodes. This leads to a huge overhead in communication. Furthermore, this information is not very reliable because of the drastic load variation in the Grid and the need for frequent updating.

Static algorithms are not affected by the system state since their behavior is predetermined. On the other hand, dynamic algorithms make decisions according to the system state. The state refers to certain types of information, such as the number of jobs waiting in the ready queue, the current job arrival rate, etc. Dynamic algorithms tend to have better performance than static ones. Some dynamic load balancing algorithms are adaptive; in other words, dynamic policies are modifiable as the system state changes. [M.Saleh, H.Deldari, B. Mokarram, 2008]

A good description of customized load balancing strategies for a network of workstations can be found in [Zaki, Li, and Parthasarathy,1996]. More recently, [Houle, Symnovis, and Wood,2002] consider algorithms for static load balancing in tree model, assuming that the total load is fixed. On the contrary, in the traditional distributed systems for which a lot of algorithms have been proposed, few have focused on grid computing. This is due to the innovation and the specific characteristics of this infrastructure.

Load balancing algorithms can be defined by their implementation of the following policies [Karatza,1994] [Anamika Jain and Ravinder Singh, 2013] [Ardhendu Mandal and Subhas Chandra Pal, 2010][A.S Manekar, M.D Poundekar, H.Gupta, M.Nagle,2012] [L.Hima,2011][32][33] [Deepika, Divya Wadhwa and Nitin Kumar, 2014]:

• *Information policy* specifies what workload information to be collected, when it is to be collected and from where.
• *Triggering policy* determines the appropriate time to start a load balancing operation.
• *Resource type policy* classifies a resource as *server* or *receiver* of tasks according to its availability status.
• *Location policy* uses the results of the resource type policy to find a suitable partner for a server or receiver.
• *Selection policy* defines the tasks that should be transferred from overloaded resources (source) to the idlest resources (receiver).

II. PROPERTIES OF DISTRIBUTED LOAD BALANCING

2.1. Global vs. Local Strategies [Ankush P. Deshmukh, K. Pamu, 2012][Ardhendu Mandal, Subhas Chandra Pal,2010][G.Sharma, J. Kaur,2013][32][33] [Deepika, Divya Wadhwa and Nitin Kumar, 2014]

Global or *local* policies answer the question of what information will be used to make a load balancing decision. In *global* policies, the load balancer uses the performance profiles of all available workstations. In *local* policies, [Ankush P. Deshmukh, K. Pamu, 2012] workstations are partitioned into different groups. In a heterogeneous NOW, the partitioning is usually done such that each group has nearly equal aggregate computational power. The benefit of a local scheme is that performance profile information is only exchanged within the group.

The choice of a global or local policy depends on the behavior of an application. For global schemes, balanced load convergence is faster compared to a local scheme since all workstations are considered at the same time.

However, this requires additional communication and synchronization between the various workstations; the local schemes minimize this extra overhead. But the reduced synchronization between workstations is also a downfall of the local schemes if the various groups exhibit major differences in performance.

[ZAKI96] notes that if one group has processors with poor performance (high load), and another group has very fast processors (little or no load), the latter will finish quite early while the former group is overloaded.

2.2 Dynamic and Static Load Balancing

A load balancer with dynamic load balancing allocates/re-allocates resources at runtime and uses the system-state information to make its decisions. Adaptive load balancing algorithms are a special class of dynamic algorithms. They adapt their activities by dynamically changing their parameters, or even their policies, to suit the changing system state. DLB is used to provide application level load balancing for individual parallel jobs. It ensures that all loads submitted through the DLB environment are distributed in such a way that the overall load in the system is balanced and application programs get maximum benefit from available resources.

Every dynamic load balancing method must estimate the timely workload information of each resource. This is the key information in a load balancing system where responses are given to following questions: (i) how to measure resource workload; (ii) what criteria are retaining to define this workload; (iii) how to avoid the negative effects of resource dynamicity on the workload; and, (iv) how to take into account the resource heterogeneity in order to obtain an instantaneous average workload representative of the system. [Ardhendu Mandal, Subhas Chandra Pal, 2010]

According to the taxonomy proposed in [Casavant and Kuhl,1988], the first distinction is between static and dynamic algorithms.

In *static* algorithms, information about the total mix of processes in the system is assumed to be known by the time the executable image of a program is linked, and this information is used to assign a processor to the program: each time the program is started, the corresponding process is run on that processor. In *dynamic* algorithms, no (or little) *a priori* information is required about resource demands of processes, and no assumption is made about what the system state will be at program execution time. When local conditions make a process migration desirable, the location policy selects a suitable machine for receiving the process. [Sherihan Abu Elenin and Masato Kitakami, 2011][Abhijit A. Rajguru, S.S. Apte,2012][[Abhijit A. Rajguru, S.S. Apte, 2012] [Deepika, Divya Wadhwa and Nitin Kumar, 2014] R.Prajapati, D.Rathod, S.Khanna, 2015] [N.Goyal, 2013].

In static load balancing, the performance of the processors is determined at the beginning of execution. Then depending on their performance, the work load is distributed in the beginning by the master processor [Baldeschwieler, Blumofe, Brewer, Atlas, 1995] [Sherihan Abu Elenin and Masato Kitakami, 2011]. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned, that is static load balancing methods are no preemptive.

The goal of static load balancing method is to reduce the overall execution time of a concurrent program while minimizing the communication delays. A general disadvantage of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution. [Sherihan Abu Elenin and Masato Kitakami, 2011]

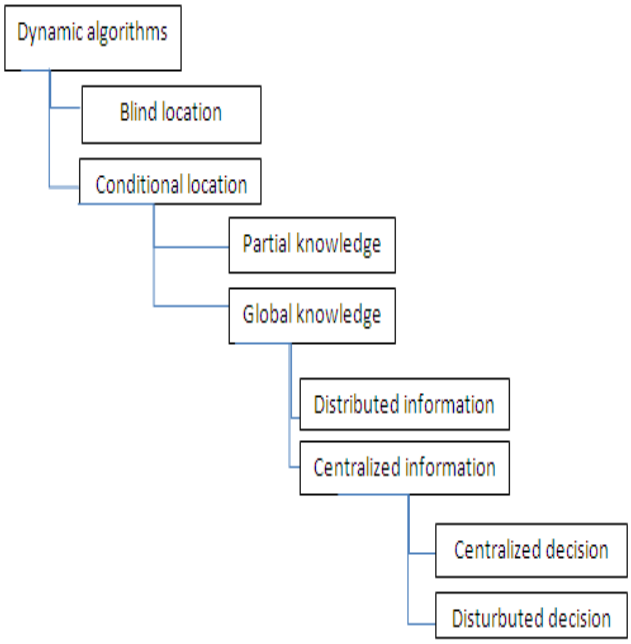

Fig.1: Dynamic policies for location and information.

The family of dynamic algorithms may be further refined (see figure 1).

In algorithms with *blind location*, the choice of an execution site is made without any information about the current conditions of the remote machines. Conversely, for *conditional location*, the choice of a receiver machine is based on a *global knowledge* or a *partial knowledge* of the system state, according to whether the decision is made with information about all the machines in the network, or about a subset. The global knowledge may be maintained on a single machine *(centralized information)* or on all the machines of the network *(distributed information)*. Finally, the migration decision may be taken by a single machine *(centralized decision)* or else may be taken by different machines *(distributed decision)*.

Like any taxonomy, this one is also not perfect, and it is difficult to find the right place for a few algorithms.

Static algorithms have two drawbacks. First, their execution cost is high; hence they cannot be used to react to fast changes in the system. Second, when the variability of execution time is taken into account, they are done with exponential assumptions (in order to be able to obtain exact results), when in fact observations on real systems invalidate these assumptions.

Static algorithms may be worthwhile for computing systems that execute periodically a set of programs with well-known behavior (e.g., real-time systems). This is clearly not the case for networks of workstations. Thus, the remainder of this section will be devoted to dynamic algorithms.

2.3 Centralized and Decentralized Load Balancing: [Ankush P. Deshmukh, K. Pamu, 2012][32]

A load balancer is categorized as either *centralized* or *distributed*, both of which define where load balancing decisions are made. In a centralized scheme, the load balancer is located in one master workstation node and all decisions are made there. In a distributed scheme, the load balancer is replicated in all workstations.

Once again, there are tradeoffs associated with preferring one location scheme over the other. [Ankush P. Deshmukh, K. Pamu, 2012]

For centralized schemes, the reliance on one central point of balancing control could limit future scalability. Additionally, the central scheme also requires an "all-to-one" exchange of profile information from workstations to the balancer as well as a "one-to-all" exchange of distribution instructions from the balancer to the workstations. The distributed scheme helps solve the scalability problems, but at the expense of an "all-to-all" broadcast of profile information between workstations. However, the distributed scheme avoids the "one-to-all" distribution exchange since the distribution decisions are made on each workstation.

In this section, the paper shows differences of load balancing algorithms that are categorized in two tables (1, 2) based on centralized and decentralized:

Table 1: Parametric Comparison of Load Balancing Algorithms (centralize)

Parameters	**Central Queue**	**Central Manager**
Overload Rejection	YES	No
Fault Tolerant	YES	YES
Forecasting Accuracy	LESS	More
Stability	SMALL	Large
Centralized/ Decentralized	C	C
Dynamic/	DY	S

Static		
Cooperative	YES	YES
Process Migration	NO	NO

There are six types of decentralized load balancing as shown in table 2: Round Robin algorithm, Randomized algorithm, local Queue, Ant colony, agent based Algorithm, and Threshold algorithm.

Table 2: Parametric Comparison of Load Balancing Algorithms (centralize)

[Sherihan Abu Elenin and Masato Kitakami, 2011][Deepika, Divya Wadhwa and Nitin Kumar,2014]

Parameters	**Round Robin**	**Random**	**Local Queue**	**Ant colony**	**agents**	**Threshold**
Overload Rejection	No	No	YES	No	YES	No
Fault Tolerant	No	No	YES	YES	No	No
Forecasting Accuracy	More	More	LESS	YES	No	More
Centralize Decentralized	D	D	D	D	D	D
Stability	Large	Large	SMALL	YES	No	Large
Dynamic/ Static	S	S	DY	DY	DY	S
Cooperative	NO	NO	YES	YES	YES	YES
Process Migration	NO	NO	YES	NO	YES	NO
Resource Utilization	LESS	LESS	MORE	MORE	LESS	LESS

Round Robin algorithm [Xu and Lau,1997][Sherihan Abu Elenin and Masato Kitakami,2011][Abhijit A. Rajguru, S.S. Apte,2012][Deepika, Divya Wadhwa and Nitin Kumar,2014] distributes jobs evenly to all slave processors. All jobs are assigned to slave processors based on Round Robin order, meaning that processor choosing is performed in series and will be back to the first processor if the last processor has been reached. Processor choosing is performed locally on each processor, independent of allocations of other processors. Advantage of Round Robin algorithm is that it does not require inters process communication. In general, Round Robin is not expected to achieve good performance. [Sherihan Abu Elenin and Masato Kitakami, 2011]

Randomized algorithm [Xu and Lau,1997] uses random numbers to choose slave processors. The slave processors are chosen randomly following random numbers generated based on a statistic distribution. Randomized algorithm can attain the best performance among all load balancing algorithms for particular special purpose applications.

In Central Manager Algorithm that was categorized in table 1 [Xu and Lau,1997], in each step, central processor will choose a slave processor to assign a job. The chosen slave processor is the processor with the least load. The central processor is able to gather all slave processors' load information; thereof the choice based on this algorithm would be possible. The load manager makes load balancing decisions based on the system load information, allowing the best decision when the process is initiated. High degree of inter-process communication could create a bottleneck state. [Sherihan Abu Elenin and Masato Kitakami, 2011]

In Threshold algorithm [Xu and Lau,1997], the processes are assigned immediately to hosts upon creation. Hosts for new processes are selected locally without sending remote messages. Each processor keeps a private copy of the system's load. The load of a processor can be characterized by one of the three levels: under loaded, medium and overloaded. Two thresholds parameters t_under and t_upper can be used to describe these levels. Under loaded: load < t_under, Medium: t_under ≤ load ≤ t_upper, and Overloaded: load > t_upper. [Sherihan Abu Elenin and Masato Kitakami, 2011]

Initially, all the processors are considered to be under loaded. When the load state of a processor exceeds a load level limit, it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

III. ACO ALGORITHMS IN GRID

ACO [Casavant and Kuhl, 1994] C. R. Barde, Snehal Kasar, Samruddhi Nikam, Shradha Shelar, Nikita Wagh,2014][S. Suryadevera, J. Chourasia, S.Rathore, A.Jhummarwala,2012] [N.Goyal, 2013] is inspired by a colony of ants that work together in foraging behavior. This behavior encouraged ants to find the shortest path between their nest and food source. Every ant will deposit a chemical substance called pheromone on the ground after they move from the nest to food sources

and vice versa. Therefore, they will choose an optimal path based on the pheromone value. The path with high pheromone value is shorter than the path with low pheromone value. This behavior is the basis for a cooperative communication. There are various types of ACO algorithm such as Ant Colony System (ACS), Max- Min Ant System (MMAS), Rank-Based Ant System (RAS) and Elitist Ant System (EAS) [Xu and Lau,1997].ACO has been applied in solving many problems in scheduling such as Job Shop Problem, Open Shop Problem, Permutation Flow Shop Problem, Single Machine Total Tardiness Problem, Single Machine Total Weighted Tardiness Problem, Resource Constraints Project Scheduling Problem, Group Shop Problem and Single Machine Total Tardiness Problem with Sequence Dependent Setup Times [Xu and Lau,1997].

A recent approach of ACO researches in the use of ACO for scheduling job in grid computing [Zaki, Li, and Parthasarathy, 1996]. ACO algorithm is used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems. In [Houle, Symnovis, and Wood, 2002], ACO has been used as an effective algorithm in solving the load balancing problem in grid computing. The process taken by ACO will consider the pheromone value which depends on the time taken by each resource to process jobs. It does not consider the capacity of resources such as their bandwidth, processor speed and load. IN [Baldeschwieler, Blumofe, Brewer, Atlas,1995], two distributed artificial life-inspired load balancing algorithm are introduced, which are ACO and Particle Swarm Optimization (PSO). In the proposed algorithm, an ant acts as a broker to find the best node in term of the pheromone value stored in the pheromone table. The node with the lightest load is selected as the best node. The position of each node in the flock can be determined by its load in PSO. The particle will compare the load of nodes with its neighbors and will move towards the best neighbor by sending assigned jobs to it. The proposed algorithm performed better than ACO for job scheduling where jobs are being submitted from different sources and different time intervals.

However, PSO uses more bandwidth and communication compared to ACO. A study in [Karatza, 1994] proposed a new algorithm that is based on an echo intelligent system, autonomous and cooperative ants. In this proposed algorithm, the ants can procreate and also can commit suicide depending on existing condition. Ant level load balancing is proposed to improve the performance of the mechanism. Ants are created on demand during their lives adaptively to achieve the grid load balancing.

The ants may bear offspring when they detect the system is drastically unbalanced and commit suicide when they detect equilibrium in the environment. The ants will care for every node visited during their steps and record node specifications for future decision making. Theoretical and simulation results indicate that this new algorithm surpasses its predecessor.

However, the pheromone values were not updated in this proposed algorithm which enables the assignment of jobs to the same resource. [S. Suryadevera, J. Chourasia, S.Rathore, A.Jhummarwala, 2012]

3.1 Software Methods

In the last twenty years, researches have been searching for the techniques to improve traditional method of ant colony and to increase its speed. By categorizing ant colony into groups and sending each group to the processors, the speed application for this beneficial serial algorithm has been optimized [Glover, Kochenberger, 2003]. This application not only improves the speed of algorithm but also presents a new derivation which has positive impact on the results. The major component is usually between the minor particles and grand particles of algorithm application.

The most classic suggestion in relation to the parallel method of application such as the applying on multiprocessors, graphic processors, and grade environments are good opportunities for parallel estimations for optimizing the ant colony results and reducing the time of application [Pedemonte, Cancela, 2010]. Various methods have been proposed which can be divided in five groups: 1. Manorial system 2. Cellular 3. Independent run 4. Multicolony 5. Compounds. [S. Suryadevera, J. Chourasia, S.Rathore, A.Jhummarwala, 2012] [.Elenin and M.Kitakami, 2011] [H.Abdul, Nasir,K.Ku, Mohammud,A.Din,2010].

ACO algorithm for load balancing in distributed systems through the use of multiple ant colonies is proposed in [Chou and Abraham, 1982]. In this algorithm, information on resources is dynamically updated at each ant movement. Load balancing system is based on multiple ant colonies information. Multiple ant colonies have been adopted such that each node will send a coloured colony throughout the network.

Coloured ant colonies are used to prevent ants of the same nest from following the same route and also enforcing them to be distributed all over the nodes in the system and each ant acts like a mobile agent which carries newly updated load balancing information to the next nodes. This proposed algorithm has been compared with the work-stealing approach for load balancing in grid computing.

Experimental result shows that multiple ant colonies work better than work-stealing algorithm in term of their efficiency.

However, the multiple ant colonies do not consider resources capacity and jobs characteristics. This can make matching the jobs with the best resources a difficult task for the scheduling algorithm. [Ali, A., M.A. Belal and M.B. Al-Zoubi, 2010]][S. Suryadevera, J. Chourasia, S.Rathore, A. Jhummarwala, 2012]

The overview of mentioned applications shows that the methods of grand particles manorial and multi-colony are more reliable. The multi-colony methods have a degree of flexibility which allows them to partake in several parallel cases without wasting their functionality. Due to their flexibility and measurability, they can be conducted in grand particles. This method has the capacity of hardware application. The functionality of the minor and major partition of manorial method stem from the value and Frequency of information that is present in each tenant. Generally, the yielded results indicate that grand partition methods are better than minor ones but when the number of tenants increase, excessive communication will attack landlord and conversions will occur in route. The researchers have suggested the solution of ace crone. The minor scale manorial method was highly proposed the application of minor scale method in Graphic processors using shared memory saved in pheromone. However, the reduction of communication between central processor and GPU must be considered. Parallel cellular method and ace crone have the highest speed and performance.

3.2. Hardware Methods

The problem of this fiscal algorithm will not be solved even if it is installed as software or a powerful microprocessor because the presence of an operator and fetch of hardware will consume time of decoding and executing.

The hardware method of multi-colony algorithms in the hardware application has high speed in comparison with software. There is no limitation of running the main ant colony algorithm for the flexibility of software but in case of hardware, it is challenging to apply hardware method for the following reasons:

1) Since the amount of pheromone and the produced random quantity need to be illustrated in decimal form regarding application limits on logic arrays, there is a problem of designation.

2) Evaporation and heuristic factor that require multiplication are not supportable by most fpga arrays.

3) Since choosing the next city based on the distribution probability factor. Total sum of the previously left factors should be estimated and for each pheromone matrix a circuit should be provided. If the number of n in the problem sources up, there will be the limitations of time and space appearing on the array surface [Diessel. Gindy, Middendorf, Guntsch,ScheuermanSchmeck,So,2002]. Large decimals and repetitions in addition to divisions, the ant colony should be changed in a way that the result is not influenced by it. Hence, in some algorithms, the heuristic factors in choosing next city are ignored instead of decimals. Definite numbers have been used for powered amounts α and β are suppositions and shift will function and will be replaced by multiplication. The entire functions resulting in decimals are limited.

3.2.1 ID Based Method [*Yoshikawa and Terai 2007****]***

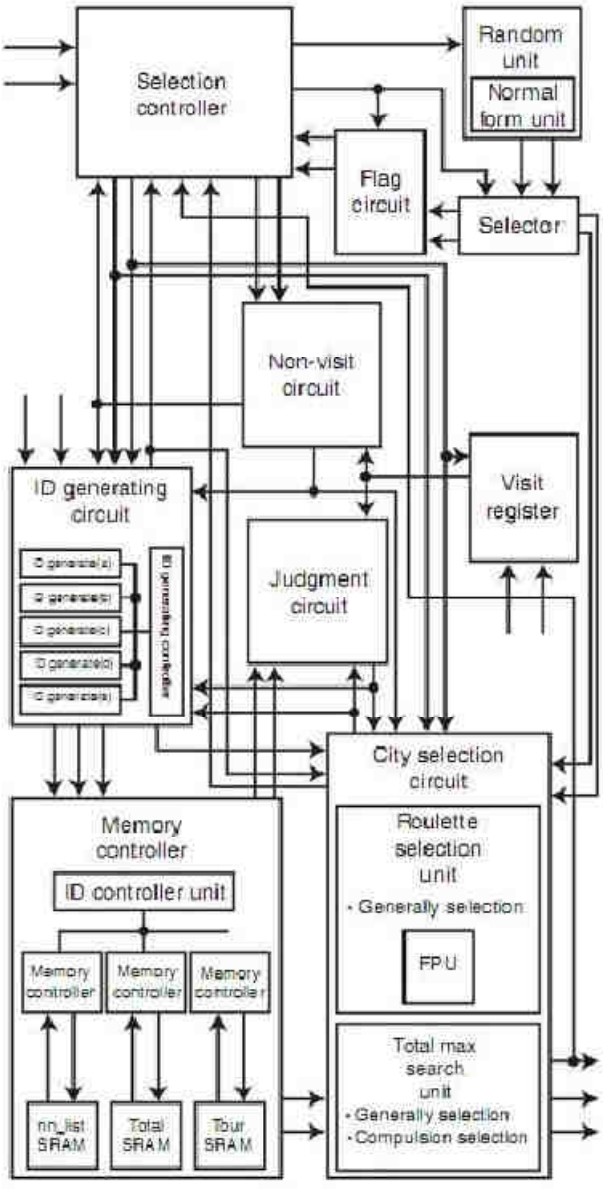

Fig.2: City Selection Diagram

Ant colony algorithm has some shortcomings:

1. Processing occurs repeatedly and sequentially
2. Pheromone variable value is very little.
3. All the ants should share pheromone variables for transition and update. Due to the repeated Processing difficulty, parallel method is perfect.

Block diagram of this method is shown as in figure 2.

Series of ids saved in SRAM memory are used for managing pheromone. They have the similar structure as the below.

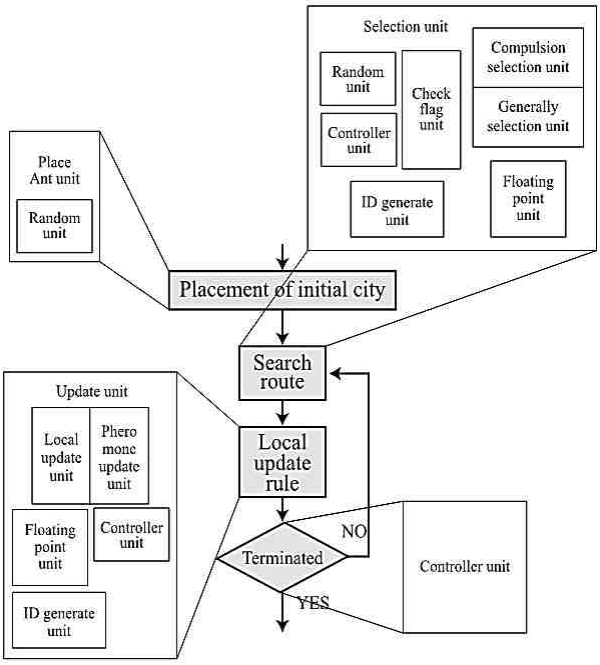

Fig.3: The relation of processing flow chart with the circuit

As in figure 3, the function of an ant using units is shown in the beginning; each ant selects randomly a node or station. After choosing the next station, each ant for each city drags with 24 bit pheromone from memory and the remaining bits which control selection and deselect of the target city uses it. Immediately after choosing city starts local updating and it does it for all the cities so that one repetition suggests one solution. In chart 3, methods of hardware are compared in a summarized form.

3.2.2 Hardware-Software Compound Method

The designation is done on the arrays in a way that C software and hardware arrays have been planned with repartition in Verilog language. The designed framework of algorithm, as shown in figure 4, is composed of two major sections. Hence, choosing the best route is done by software and is applied in C language on the hidden processors NIOSп. Other algorithm calculations are applied by hardware on array logic which can be reprogrammed. The proposed solution brings a kind of negotiation between hardware speed designation and the flexibility of software planning [Hao Yang,Wei WENG, Chen.Y,2012].

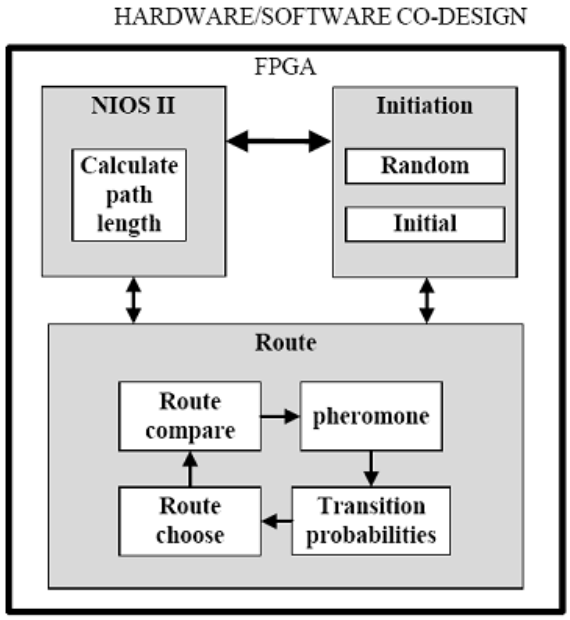

Fig.4: Ant colony circuit chart

3.2.3 Address based method [*Shafigh Fard, Monfaredi, Nadimi,2014]*

Modifications on the Ant Colony Algorithm: Some modifications have been conducted on the algorithm to reduce hardware costs; for example, simple register shift operations have been used instead of multiplication operations as well as a series of simple changes have been conducted in the update process without defecting the main program to prevent numbers from being displayed as decimal ones.

The architectural structure: The framework designed for the algorithm has consisted of a reconfigurable chip so that the ant colony parameters are defined in two blocks of memory on reconfigurable chip and all arithmetic operations of the algorithm are hardware modeled on FPGA logics. Figure 5 shows the presented framework and the connections between different blocks; as shown in the figure, there are two independent memories including main parameters of the algorithm along with evaluation, update, and city selection units each of which has consisted of a series of sub-blocks.

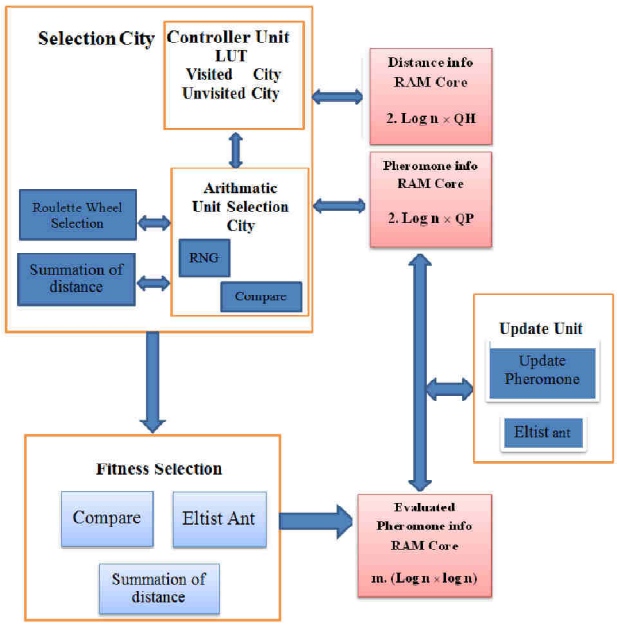

Fig.5:The architectural structure of the ant colony algorithm based on the programmable system-on-chip [Shafigh Fard, Monfaredi, Nadimi,2014]

The next city selection block: As shown in figure 3, this block consists of two blocks as 1- the control unit and 2- the arithmetic unit including sub-blocks for generating random numbers, comparison, and a series of arithmetic operations based on the Roulette Wheel law. In the following, the blocks and their functions are discussed in detail.

The control unit: As mentioned in section 2, ants respectively and node to node carry out traversal operations from the nest based on the laws of probability to reach the food; however, to prevent the repetitive selection of a node in the path, it should be controlled that the traversed nodes are separated from the ones not traversed. *[Shafigh Fard, Monfaredi, Nadimi, 2014]*

Inter-chip memories: have been selected and the hardware core of the ant colony optimization algorithm has been modeled on FPGA logics. The memory of the control unit which has been simulated as a multiplexer uses LUTs distributed in FPGA platform; this N*1 multiplexer is firstly initialized as n 0-bit inputs to convey the concept of deselecting the desired city, which takes the flag with value 1 in the multiplexer after repeating the operation and meeting the condition of selecting the relevant node.

The control unit: As mentioned in section 2, ants respectively and node to node carry out traversal operations from the nest based on the laws of probability to reach the food; however, to prevent the repetitive selection of a node in the path, it should be controlled that the traversed nodes are separated from the ones not traversed.

IV. DISCUSSION

Most of approaches based on centralized and decentralized load balancing methods were reviewed, and some important techniques were categorized in five groups. Then their advantages and disadvantages were explained and compared to each other. At last, swarm method was reviewed and the result was that this method with good properties has more advantages in comparison to other techniques.

V. CONCLUSION

For clear of operation software and hardware, table 3 shows speed of running ant colony in platform hard and soft which has been discussed in previous section.

Table.3: Speed Methods Compared

Methods name	SPEED UP
Architect for high speed ant colony optimization(ID BASED)	6.2
HARDWARE-SOFTWARE	7
Address based	110
G.P.U with 8 Cored	7.30
G.P.U with 4 cored	3.65
G.P.U with 2 cored	1.89

As is shown in the table 3, speed up of hardware methods are more than software methods, because software methods run in CPU which has a processor in compare to hardware devices that have a lot of process elements to parallel process applications.

VI. ACKNOWLEDGEMENT

I would like to express my very great appreciation to Dr. Safi and Dr. Nadimi, for their valuable and constructive suggestions during the planning and development of this research work. Their willingness to give their time so generously has been very much appreciated. Lastly, I wish to thank my parents for their support and encouragement throughout my study.

VII. AUTHOR'S CONTRIBUTIONS

This paper is resulting work of elnaz shafigh fard.

VIII. ETHICS

There isn't any ethical problem because this work is survey of load balancing and different methods of it and there isn't any work similar to this work and no other author is in this paper.

REFERENCES

[1] R.U. Payli, E. Yilmaz, A. Ecer, H.U. Akay, and S. Chien. A Dynamic Load Balancing Tool for Grid Computing. *Parallel CFD Conference*, Grand Canaria, Canary Islands, Spain, May 24-27, 2004.

[2] Ardhendu Mandal, Subhas Chandra Pal. An Empirical Study and Analysis of the Dynamic Load Balancing Techniques Used in Parallel Computing Systems *in the Proceedings of ICCS-2010*, 19-20 Nov, 2010

[3] Anamika Jain and Ravinder Singh." Review of Peer to Peer Grid Load Balancing Model Based on Ant Colony Optimization with Resource Management". *International Journal of Advanced Research in Computer Science and Software Engineering.pp 503-507.2013*

[4] A.S Manekar, M.D Poundekar, H.Gupta, M.Nagle. A Pragmatic Study and Analysis of Load Balancing Techniques in Parallel Computing. *International Journal of Engineering Research and Applications.* Vol. 2, Issue4, July-August 2012, pp.1914-1918.

[5] Abhijit A. Rajguru, S.S. Apte. A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters. *International Journal of Recent Technology and Engineering.* Volume-1, Issue-3, August 2012.

[6] Ankush P. Deshmukh, K. Pamu. Applying Load Balancing: A Dynamic Approach. *International Journal of Advanced Research in Computer Science and Software Engineering.* Volume 2, Issue 6, June 2012.

[7] Sherihan Abu Elenin and Masato Kitakami, Performance Analysis of Static Load Balancing in Grid, *International Journal of Electrical & Computer Sciences IJECS-IJENS* Vol: 11 No: 03.june 2011.

[8] Deepika, Divya Wadhwa and Nitin Kumar. Performance Analysis of Load Balancing Algorithms in Distributed System. *Advance in Electronic and Electric Engineering*. ISSN 2231-1297, Volume 4, Number 1 (2014), pp. 59-66.

[9] C. R. Barde, Snehal Kasar, Samruddhi Nikam, Shradha Shelar, Nikita Wagh. ANT Algorithm for Load Balancing Problem in FTP Server. *International Journal of Emerging Technology and Advanced Engineering.* Volume 4, Issue 2, February 2014.

[10] Ali, A., M.A. Belal and M.B. Al-Zoubi, 2010. Load balancing of distributed systems based on multiple ant colonies optimization. *Am. J. Applied Sci.*, 7: 428-433.

[11] T.L. Casavant and J.G. Kuhl. A taxonomy of scheduling in general purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–153, 1994.

[12] M. Houle, A. Symnovis, and D. Wood. Dimension-exchange algorithms for load balancing on trees. In *Proc. of 9th Int. Colloquium on Structural Information and Communication Complexity*, pages 181–196, Andros,Greece, June 2002.

[13] H.D. Karatza. Job scheduling in heterogeneous distributed systems. *Journal. of Systems and Software*, 56:203–212, 1994

[14] Baldeschwieler, J. E, Blumofe, R.D, Brewer, E.A (1995) Atlas: An Infrastructure for Global Computing. Proc. of HPCN'95, High Performance Computing and Networking Europe. Springer, Berlin Heidelberg, pp. 582-587 .

[15] C.Z. Xu and F.C.M. Lau. *Load Balancing in Parallel Computers: Theory and Practice*. Kluwer, Boston, MA, 1997

[16] M.J. Zaki, W. Li, and S. Parthasarathy. Customized dynamic load balancing for a network of workstations. In *Proc. of the 5th IEEE Int. Symp. HDPC*, pages 282–291,*1996*

[17] Casavant T L and Kuhl **1** *G* 1988 A taxonomy of scheduling in general-purpose distributed computing systems ***IEEE Trans. Sofrware Ens.*** **SE-14** 141-54.

[18] F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, International Series in Operations Research & Management Science, 57, Springer, 2003.

[19] M. Pedemonte, H. Cancela, A cellular ant colony optimisation for the generalized Steiner problem, *International Journal of Innovative Computing and Applications 2 (3) (2010) 188–201.*

[20] Chou T and Abraham **J** A 1982 Load balancing in distributed systems **IEEE Trans. Software Ens. SE4** 401-12.

[21] Diessel.H.ELGindy,M.Middendorf,M.Guntsch,B.Scheuermann,H.Schmeck,K.So,"Population Based ant colony optimization on FPGA",in:IEEE International conference on Field-Programmable Technology(FPT),2002,PP.125-132.

[22] Yoshikawa, M and Terai,H 2007 : Architecture for high speed ant colony optimization .oroceedings of IEEE International Conference on information Reuse and integration ,lasVegas,NV, 1-5

[23] Li.S,Hao Yang.M ,Wei WENG.CH,Hong Chen.Y, Ant Colony Optimization design and its FPGA implementation,.IEEE International Symposium on Intelligent Signal Processing and Communication system PP 262-265,November 4-7,2012.

[24] Elnaz Shafigh Fard, Khalil Monfaredi, Mohammad H. Nadimi, AN AREA-OPTIMIZED CHIP OF ANT COLONY ALGORITHM DESIGN IN HARDWARE PLATFORM USING THE ADDRESS-BASED METHOD, *International Journal of Electrical and Computer Engineering* , Vol 4, No 6: December 2014.

[25] MOHSEN AMINI SALEH, HOSSEIN DELDARI AND BAHARE MOKARRAM DORRI, " Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants, "Informatics, vol. 32, Pages 327–335, 2008.

[26] Lalitha Hima, Venkatesan, Karunya university, coimbatore, India "Perspective Study on Resource level Load balancing in Grid Computing Environments" 2011 IEEE.

[27] Gaurav Sharma, Jagjit Kaur Bhatia, A Review on Different Approaches for Load Balancing in Computational Grid in *Journal of Global Research in Computer Science*, **Volume 4, No. 4, April 2013.**

[28] Ramesh Prajapati, Dushyantsinh Rathod, Samrat Khanna, COMPARISON OF STATIC AND DYNAMIC LOAD BALANCING IN GRID COMPUTING , *International Journal For Technological Research In Engineering*, Volume 2, Issue 7, March-2015.

[29] Nandita Goyal, Dynamic Load Balancing Algorithm for Cloud Computing using Mobile Agents, International Journal of *Advanced Research in Computer Science and Software Engineering*, Volume 3, Issue 12, December 2013.

[30] Sowmya Suryadevera, Jaishri Chourasia, Sonam Rathore, Abdul Jhummarwala, Load Balancing in Computational Grids Using Ant Colony Optimization Algorithm, *International Journal of Computer & Communication* Technology (IJCCT) ISSN (ONLINE): 2231 - 0371 ISSN (PRINT): 0975 –7449 Vol-3, Iss-3, 2012.

[31] Husna Jamal Abdul Nasir,Ku Ruhana Ku Mohammud,Aniza Mohamed Din, load balancing using enhanced ant algorithm in Grid computing, second international conference on computational intelligence, modeling and simulation,2010 IEEE.

[32] Essays, UK. (November 2013). Dynamic Load Balancing Algorithms Computer Science Essay. Retrieved from https://www.ukessays.com/essays/computer-science/dynamic-load-balancing-algorithms-computer-science-essay.php?cref=1

[33] Essays, UK. (November 2013). Load Balancing Algorithms In Grid Environment Information Technology Essay. Retrieved from https://www.ukessays.com/essays/information-technology/load-balancing-algorithms-in-grid-environment-information-technology-essay.php?cref=1

[34] Javier Bustos-Jimenez, Denis Caromel and Jose M.Piquer, Load Balancing: Toward the Infinite Network and Beyond,springer 2007.